The Compact Muon Solenoid Experiment
TriDAS Trigger and Data Acquisition

RCMS
User's Manual

Version 1.0
11/4/03

Version:	1.0
Date:	11/3/03
Author:	M. Gulmini
CI Identifier:	rcms_um_D_19673

European Organization for Nuclear Research
Organisation Européenne pour la recherché nucléaire
1211 Geneva 23, Switzerland

Revision History

Date	Version	Description	Author
28/10/03	1.0	Document creation	M. Gulmini

Table of Contents

1 Introduction

The CMS data acquisition group is presently developing a Run Control and Monitor System as described in the TriDAS TDR [1]. This document tells how to obtain, install, build and operate the software made available as version 1.0.

The main goal of this document is to make the reader able to install, configure, operate and extend the Run Control software for use in CMS application scenarios. To achieve this goal some examples are provided.

It is not the purpose of this document to describe the internals workings of the software.

This first chapter provides an overview of the Run Control as described in the TriDAS TDR.

1.1 RCMS Overview

RCMS (Run Control and Monitor System) is defined as the collection of hardware and software components responsible for controlling and monitoring the CMS experiment during data-taking. The RCMS has to provide users with a "virtual counting room", enabling them to operate the experiment from anywhere in the world.

The RCMS views the experiment as a collection of partitions. A partition is a configurable group of resources. Multiple partitions can be active concurrently, allowing several sections of the experiment to run independently.

A number of DAQ sub-systems are involved in data-taking (see figure 1). Any sub-system can be partitioned and different partitions can be operated concurrently.

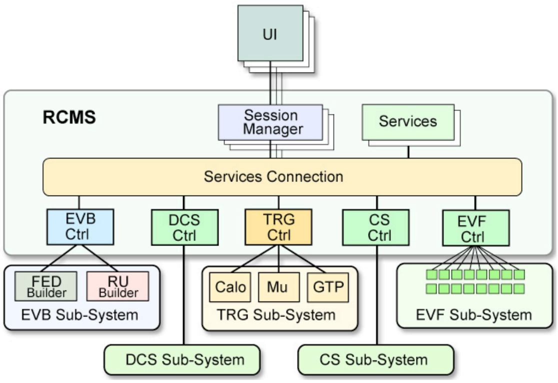


Figure 1: Session Managers and Sub-Systems defined in the RCMS

The running of a partition is defined as a "session". Each session is associated with a Session Manager (SMR) that coordinates user access to the session and propagates commands from the users to the Sub-System Controllers (SSCs). Multiple sessions may coexist and run concurrently.

A Sub-System Controller consists of a number of Function Managers (FMs) and local services.

One Function Manager is instantiated for each sub-system partition. A FM receives requests from the corresponding SMR and transforms them into commands for the sub-system resources. The results of performed actions are logged and analyzed by the RCMS. A number of services support the interaction with the user to manage sub-system resources.

Figure 2 shows a block diagram of the RCMS with the services defined so far.

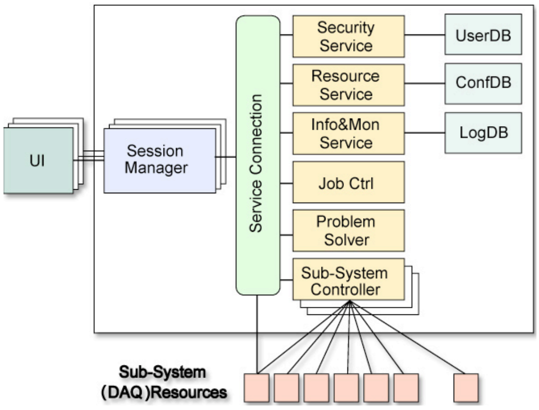


Figure 2: Block Diagram of the Run Control and Monitor System

The Security Service provides facilities for user authentication and authorization, including data encryption when demanded. It manages the user database where profiles are stored, including access rights and a working history.
The Resource Service handles all the resources of the DAQ, including partitions. A resource can be any hardware or software component involved in a DAQ session. Resources can be discovered, allocated and queried. Partitions can only use available resources.
The Information and Monitor Service collects messages and monitor data coming from DAQ resources and RCMS components and stores them in a database. There are several types of messages collected from the sub-systems. The messages are catalogued according to their type, severity level and timestamp. The Information and Monitor Service can publish the information to subscribers. These subscribers can register for specific messages categorized by a number of selection criteria, such as timestamp, information source and severity level.
The Job Control is a utility that allows the remote execution and supervision of any software process involved in data-taking. It is used to start, stop and monitor the software infrastructure of the RCMS and the DAQ software components.
The Problem Solver has to identify malfunctions of the DAQ system and determine possible recovery procedures. It subscribes to the IMS to receive the information it is interested in. The information is processed by a correlation engine and the result is used to determine a potential recovery action.

More information about the RCMS can be found in [1] and [2].

1.2 Software technologies

Most Internet applications have adopted Web technologies and in particular Web Services when distributed systems need to be interconnected. The XML data format is commonly used for data exchange and the SOAP communication protocol for communication. The use of standard or widely adopted technologies allows for maximum profit of advances made in the Internet world.
Web technologies and related developments play a fundamental role in the implementation of the RCMS. The software release described in this document is fully developed in Java and adopts the Sun Java Web Services Developer Pack (JWSDP), an integrated development tool for Web Services.
Databases also play a key role. Both native XML database technologies, based on the XMLDB interface, and relational database management systems have being investigated. While relational DBMS are a mature technology, XML native databases are an emerging technology that would remove the need to transform between XML data and relational table structures. The present release provides support for eXist native XML database and for mySQL database, that, for reliability and performance issues, is the recommended one.

1.3 References

[1] The CMS collaboration, The Trigger and Data Acquisition Project, Volume II, Data Acquisition & High-Level Trigger, CERN/LHCC 2002-26, ISBN 92-9083-111-4

[2] V. Brigljevic et al., “Run Control and Monitor System for the CMS Experiment”, Proceeding of CHEP 2003, San Diego, USA

2 RCMS Release 1.0

The main goal of the release described in this document is to provide a Run Control and Monitor System framework for XDAQ based CMS Data Acquisition Systems, like the ones used in beam tests, production and validation centers, and DAQ demonstrators.

The use of the RCMS software in several application scenarios of the CMS experiment will help understanding the requirements of the several sub-systems, to define the interfaces to them, and the relationships between the many databases involved in the CMS data-taking.
It must so be emphasized that the system is still under development and subject to change.

2.1 What is Available?

The present release provides several of the services described in the overview.
A basic Security Service, with login and password mechanism, the Resource Service, the Information and Monitor Service, and the Session and Function Managers.
A Job Control service and the Problem Solver are not released yet, and will be added in future releases.
The Resource Service provides the user with an interface to store DAQ configuration information in a database.
An applet based Graphical User Interface (GUI) allows for insertion and retrieval of data. The GUI is developed with the aim of facilitating the setup and configuration of any XDAQ based data acquisition system. The GUI is extensible, so that specific displays can be developed as separate components by the users and plugged in the GUI framework.
Session and Function Managers provide the engines for controlling XDAQ applications. Based on Finite State Machines they can be customized by the user in order to provide the specific behaviours needed to control a specific sub-system.
The Information and Monitor Service allows for collecting information coming from any component involved in a data acquisition session, including other RCMS services, and storing such information in a database. It implements a publisher-subscriber protocol for clients that need to be notified when the information they are interested is available. The release provides the source code and the XML schema of the current IMS prototype, but no GUIs or examples are included. They will be provided with the next RCMS release.
The services can be configured to use the eXist native XML database or the mySQL database. The Information and Monitor Service provides also support for flat files.

The next sections will guide you through the installation and your first steps with RCMS.

3 Getting Started

3.1 System Requirements and External Packages

RCMS is developed using the Java programming language. The code has been written and tested using the Java Development Kit 1.4 (JDK1.4). The Linux operating system running on an Intel x86 processor has been used. The Graphical User Interface has also been tested on Windows machines where a JDK1.4 was installed.

JDK1.4 must be installed on the Linux host where you install the RCMS release. JDK1.4 must also be installed on the remote machines where you want to use the GUI. Previous Java versions will not work.

RCMS currently supports XDAQ version 1.2 and 1.3.

Since the RCMS services are developed as Java servlets, they need to be deployed in a servlet container. The recommended container is the Apache Jakarta Tomcat 4.

The services make use of database management systems. EXist native XML database and mySQL are currently supported. We recommend the use of mySQL.

To summarize, the present RCMS release needs the following external packages:

Java JDK1.4 (http://java.sun.com)
Apache Jakarta Tomcat 4.0 (http://jakarta.apache.org)
MySQL database (http://www.mysql.com) or
eXist database 0.9 (http://exist.sourceforge.net)

3.2 Installation

The Run Control and Monitor System release 1.0 is available as a tar gzipped file, containing the source code, the Jakarta Tomcat servlet container and the eXist 0.9 native XML database. You can download it from the XDAQ web site. The file does not contain the mySQL database and the Java JDK1.4.

MySQL and JDK1.4 must be installed as separate packages.

The source code is also available in the CERN TriDAS CVS server.

The following paragraph explains how to checkout the RCMS Release 1.0 from the CVS server.

It is assumed that the user shell is tcsh, and that the installation directory is **InstallationDir** (please replace **InstallationDir** with the full path to your installation directory).

3.2.1 Downloading the source code

Set the environment variable **CVSROOT** to point to the CMS CVS server.

If you are at CERN:

```
> setenv CVSROOT :kserver:cmscvs.cern.ch:/cvs_server/repositories/TriDAS
```

If you are outside CERN you can use the anonymous login:

```
> setenv CVSROOT :pserver:anonymous@cmscvs.cern.ch:/cvs_server/repositories/TriDAS
> cvs login
```

using the password **98passwd**

Go to the installation directory.

```
> cd InstallationDir
```

Checkout release 1.0 of the RCMS source code.

```
> cvs co –rrcms_G_19674_V1_0 TriDAS/daq/services/rcms
```

The directory **InstallDir**/TriDAS/daq/services/rcms is referred as **RCMSDir.**

3.2.2 *Source code Directory Structure*

The directory **RCMSDir** contains one subdirectory for each service.

rs:	Resource Service (plus basic Security Service)
ims:	Information and Monitor Service
manager:	Session and Function Managers
gui:	Graphical User Interface

The source code for each service is then located in the directory src/java.

3.2.3 *Configuring RCMS*

In order to compile successfully the RCMS source code you need to modify a few configuration parameters.
Since mySQL is the recommended database the RCMS distribution is configured to use it by default.
It is supposed that the mySQL DB is installed on the same host as RCMS.
The default mySQL user is 'rcms', without password, and two databases named 'rs' and 'ims' must already exist.
The default configuration also foresees to deploy the RCMS services in a servlet container accessible on port 8080.
The default configuration can be changed editing a couple of configuration files.
The Resource Service configuration file

RCMSDir/rs/src/java/rcms/rs/**RSConfiguration.java**

And the Information and Monitor Service configuration file

RCMSDir/ims/src/java/rcms/ims/**IMSConfiguration.java**

Both the configuration files contain the comments necessary to modify them in the correct way.

The RCMS source code will be compiled using the Ant Build Tool. The file **build.xml** used by the Ant Build Tool is located in the directory **RCMSDir.** If you don't use the servlet container provided with the distribution tar file, you need to edit and modify the build.xml file. In particular the entry **jakarta.dir** has to contain the path of your Jakarta installation. For instance:

```
<property name=”jakarta.dir” value=”/home/rcms/jakarta-tomcat-4.1.18”/>
```

3.2.4 Compiling the source code

Once it has been configured the source code is ready to be compiled. The Ant Build Tool is used. Ant binaries and libraries have already be downloaded from the cvs server.

Go to the RCMS directory

```
> cd RCMSDir
```

Compile and deploy RCMS

```
> bin/ant install
```

The command compiles all the source code, creates the deployment structure, and copies it in the proper directory of the Jakarta servlet container.
If you would like to clean the results of the compilation you can use

```
> bin/ant clean
```

The command

```
> bin/ant
```

compiles everything without deploying the services in the Jakarta tomcat container.

It is also possible to create a Javadoc documentation of the APIs used by the services.

```
> bin/ant doc
```

Producing the javadoc documentation you might see many error messages related to external libraries, just ignore them.

3.3 Starting the RCMS services

The procedures to startup and shutdown the RCMS services are provided by the Jakarta Tomcat package.
If you downloaded the tar distribution file the **JakartaDir** directory is **InstallDir**/jakarta-tomcat-4.1.18-LE-jdk14.
Go to the Jakarta installation directory

```
> cd JakartaDir
```

Startup the services

```
> bin/startup.sh
```

If necessary all the services can be shut down:

```
> bin/shutdown.sh
```

3.4 Launching the Client GUI

Once the RCMS services have been started they are ready to accept SOAP over HTTP requests from the clients.
A client graphical user interface (GUI) framework based on Java applets has been developed.
The GUI provides a user interface to the Resource Service and includes some displays for commanding and monitoring XDAQ based systems. Users can extend the GUI developing new applets and plugging them into the GUI framework. Refer to Chapter 6 “How to Extend the GUI” for more details.

You can launch the GUI on any host where a JDK1.4 is installed by using the **appletviewer.**
Also web browsers where a java plugin 1.4 is installed can be used.
In both cases the Java Security Policies must be correctly configured. Refer to the Java documentation or to your browser documentation for the correct configuration.
The easiest way to start with RCMS and verify if your installation is correct is to use the **appletviewer** on the same Linux host where RCMS is installed.

Put a file named **.java.policy** in your home directory with the following content:

```
grant { permission java.security.AllPermission; };
```

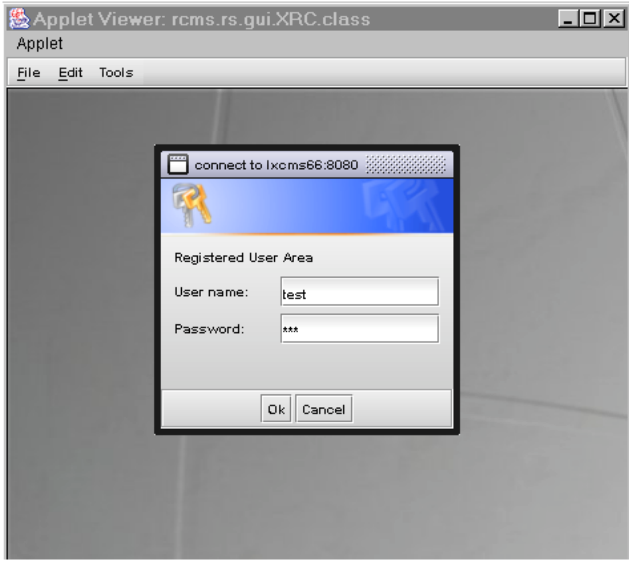
Now you can launch the GUI

```
> appletviewer http://<hostname>:<port>/RCMS/GUI.html
```

For instance

```
> appletviewer http://localhost:8080/RCMS/GUI.html
```

You will see the applet in Figure 3:



Registered User Area
User name: test
Password: ***

Figure 3: RCMS GUI - login

Now try to login as user ‘guest’ and password ‘xxx’.
The login window disappears and you are ready to play with your first RCMS example.

4 Your First Example

This chapter explains how to insert your first configuration into the Resource Service in order to control and monitor your first XDAQ based system via RCMS.

4.1 Installing the Example

Once you have installed RCMS the first time, the Resource Service database is empty. A program that fills the database with a simple configuration is provided.

Go to the Resource Service ‘samples’ directory

```
> cd RCMSDir/rs/src/java/rcms/rs/samples
```

Run the shell script ‘filltestdb.sh’

```
> ./filltestdb.sh
```

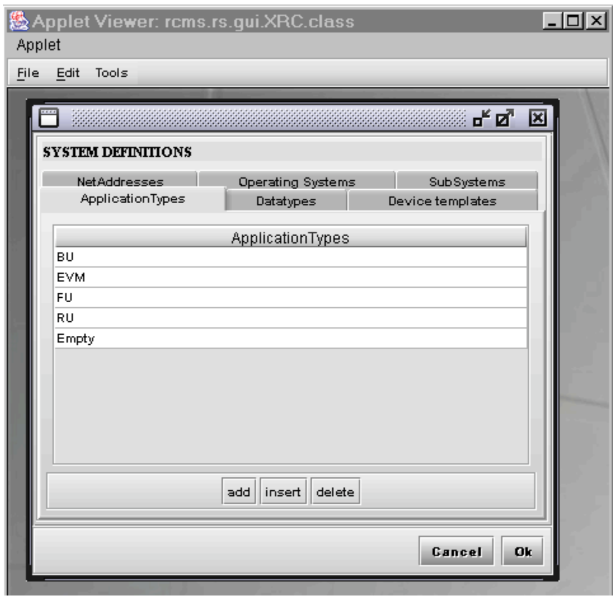
The script starts a Java application (FillTestDB.java) that creates a new RCMS user (user ‘test’, password ‘xxx’) and inserts a simple configuration into the Resource Service. A partition containing two XDAQ ‘Empty’ applications is created. The source code of the Empty XDAQ application is located in the directory

```
> RCMSDir/manager/src/xdaq/EmptyApp
```

4.2 Browsing The Configuration

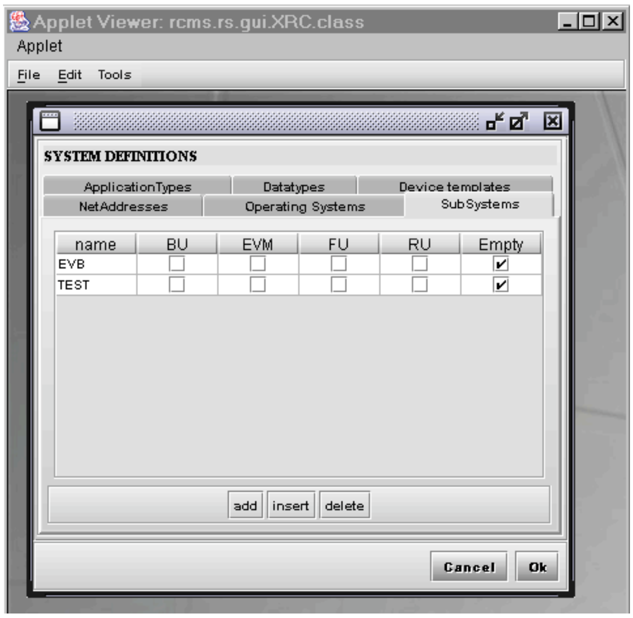
In this paragraph we will have a look to the example configuration using the GUI.
You must login to the RCMS using the user **test** with password **xxx**.

Choose **System Definitions** in the **Edit** menu.
The first step is to define the XDAQ application types,for instance the **Empty** application type (figure 4).
The Empty application type is then declared as an application belonging to the **TEST** sub-system (figure 5).



ApplicationTypes
BU
EVM
FU
RU
Empty

Figure 4: System Definitions: application types



name	BU	EVM	FU	RU	Empty
EVB	□	□	□	□	☑
TEST	□	□	□	□	☑

Figure 5: System Definitions: Sub-Systems and application types

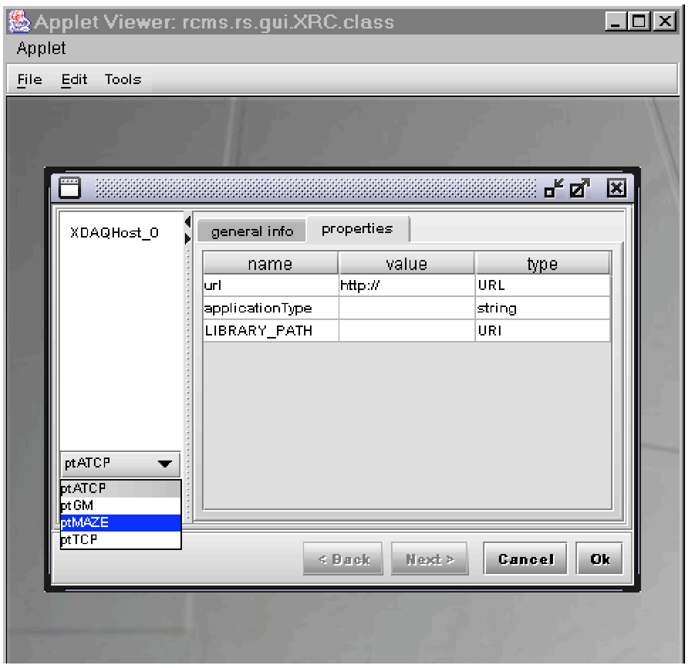
Now we can define the information about our XDAQ Hosts (figure 6). Use the **Devices** entry in the Edit menu. Two XDAQ hosts have been inserted. Click the modify button in order to see the information about a host.

The field **url** defines the XDAQ SOAP url.

The field **ApplicationType** defines the application types that can run in that host.

The field **LIBRARY_PATH** defines the common path where all the libraries that have to be used by the host are located.

Both ApplicationType and LIBRARY_PATH can be left empty, but their use provides help for an easier and clearer insertion of the other data needed for a complete configuration. Here you can also define the transports used by the host.



name	value	type
url	http://	URL
applicationType		string
LIBRARY_PATH		URI

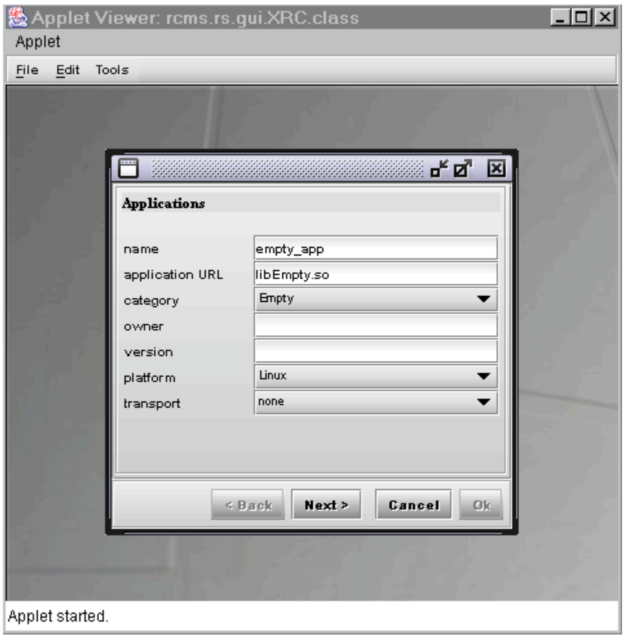
Figure 6: Defining the XDAQ Hosts

In the Edit menu, choosing **Applications** or **Transports**, you can define the XDAQ applications and the XDAQ transports. Figure 7 shows the information for the Empty application.

The **application URL** contains the name of the library providing the application's code.

If the path is not absolute the LIBRARY_PATH of the XDAQ Host where the application is loaded will be used.

For each application and transport you can also define the Default configuration Parameters (see XDAQ Companion – Configuration).



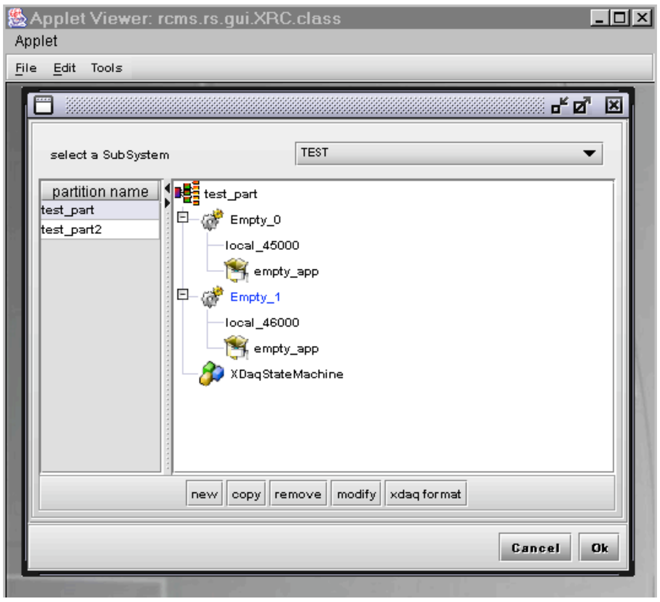
Applications

name	empty_app
application URL	libEmpty.so
category	Empty
owner	
version	
platform	Linux
transport	none

Figure 7: An Application

Once your XDAQ Hosts, your Applications, and your Transports have been defined you can create a Partition. At the purpose select **Partitions** from the Edit menu.

Any partition belongs to a Sub-System. In figure 8 a partition of the sub-system TEST, composed of two applications is shown.



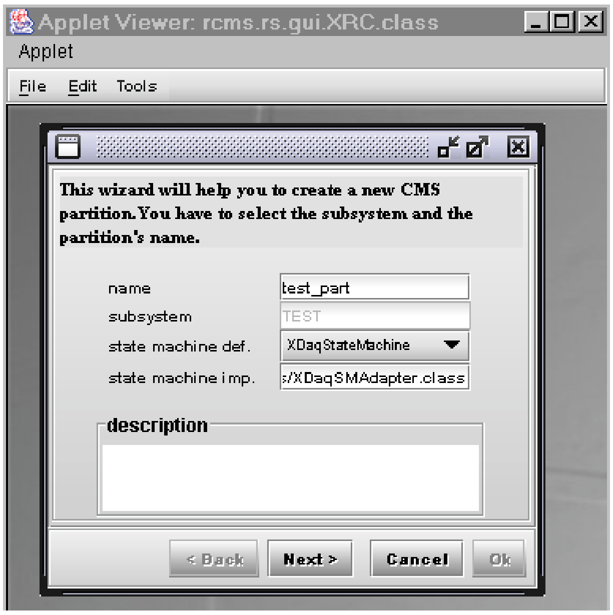
select a SubSystem: TEST

partition name
test_part
test_part2

- test_part
  - Empty_0
    - local_45000
      - empty_app
  - Empty_1
    - local_46000
      - empty_app
  - XDaqStateMachine

Figure 8: An example of Sub-System Partition

When a sub-system partition will be made active, the RCMS creates a Function Manager fully dedicated to the partition. The Function Manager behaviour can be adapted to the specific requirements of your partition. For more details see chapter 6. The url where the FM User class is located must be set. By default the class XDaqSMAdapter is used (figure 9).



This wizard will help you to create a new CMS partition. You have to select the subsystem and the partition's name.

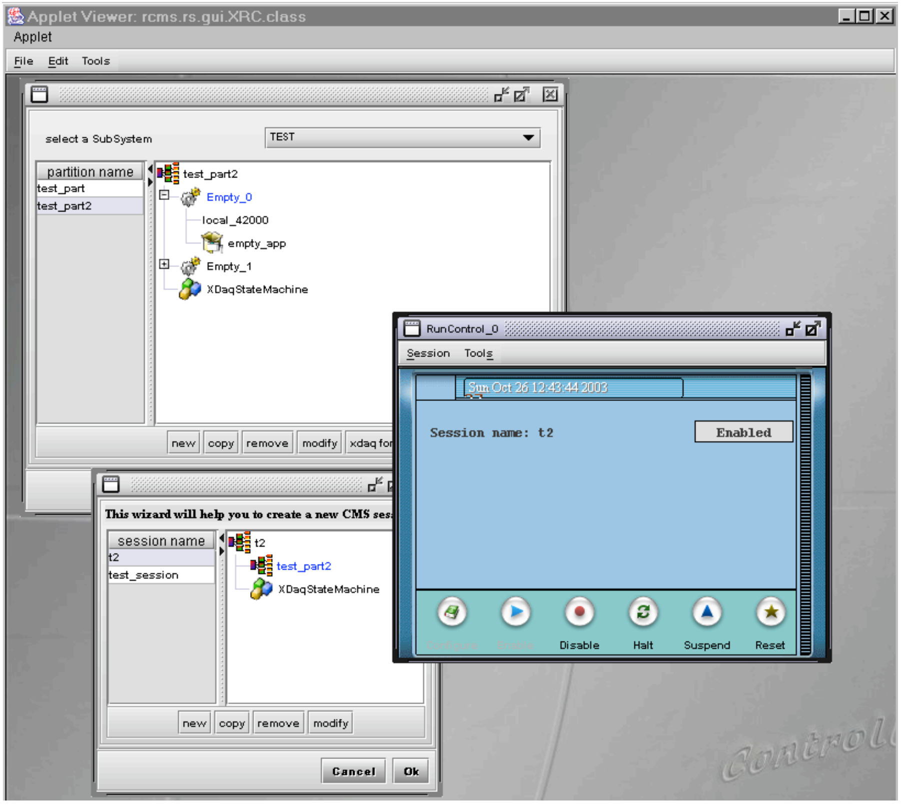
name: test_part
subsystem: TEST
state machine def.: XDaqStateMachine
state machine imp.: :/XDaqSMAdapter.class

Figure 9: Function Manager User class

To fully define your partition you must then associate the applications and the transports with the XDAQ Hosts.

As a final step you need to define which sub-system partitions will be used in your control **Session**. Figure 10 shows a **Session** named **t2** that just uses the **test_part2** partition of the TEST sub-system.

Figure 10 also shows the Run Control window, argument of the next paragraph, 'Controlling XDAQ applications'.



This wizard will help you to create a new CMS ses

Session name: t2

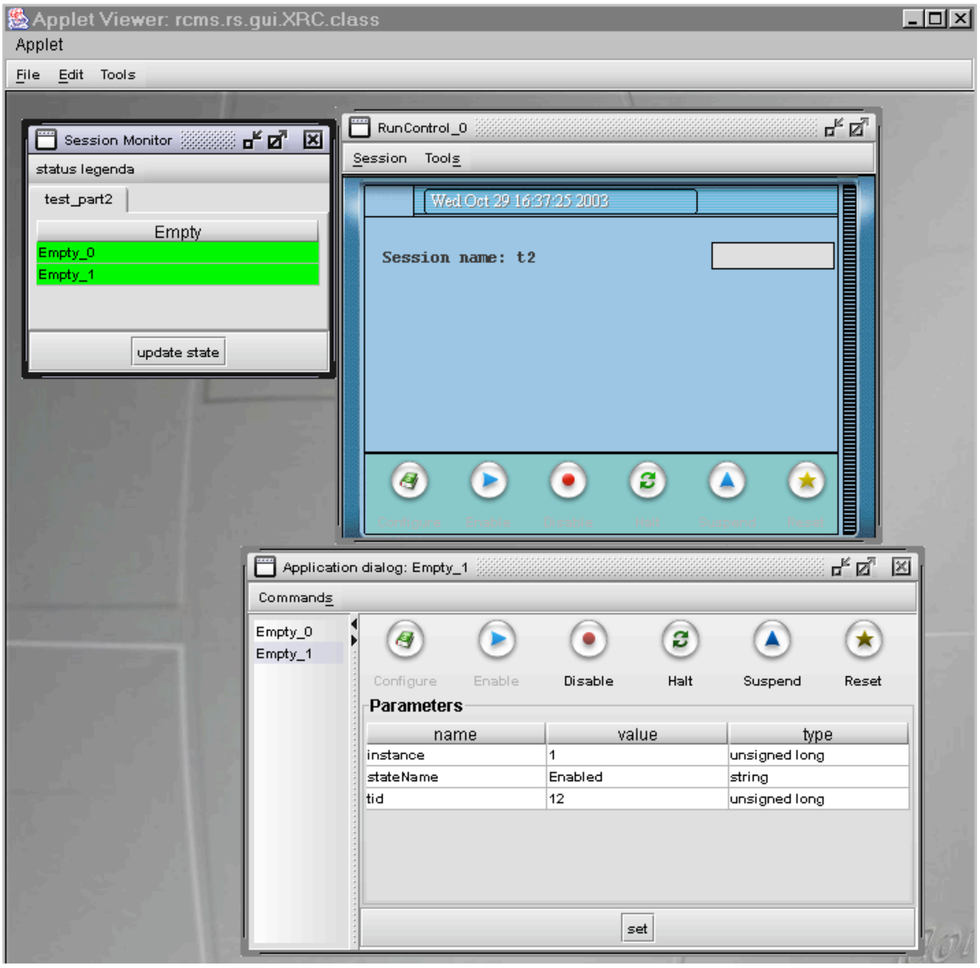
Figure 10: A Control Session

4.3 Controlling XDAQ applications

The **Tools** menu of the GUI framework provides three applets that can be used for control purposes. These applets, called **RClets**, are developed as described in chapter 5.
Users can easily modify these applets or develop new ones by following the rules described in chapter 5.

4.3.1 The RunControl applet

The **RunControl** applet provides an example of graphical user interface for controlling a XDAQ based system (see figure 11).
The applet works as a client of the RCMS Session and Function Managers.
It allows for sending commands to the applications (RunControl window), for retrieving and setting the parameters exported by the XDAQ applications (ApplicationDialog window), and for monitoring the status of the applications (SessionMonitor window).



name	value	type
instance	1	unsigned long
stateName	Enabled	string
tid	12	unsigned long

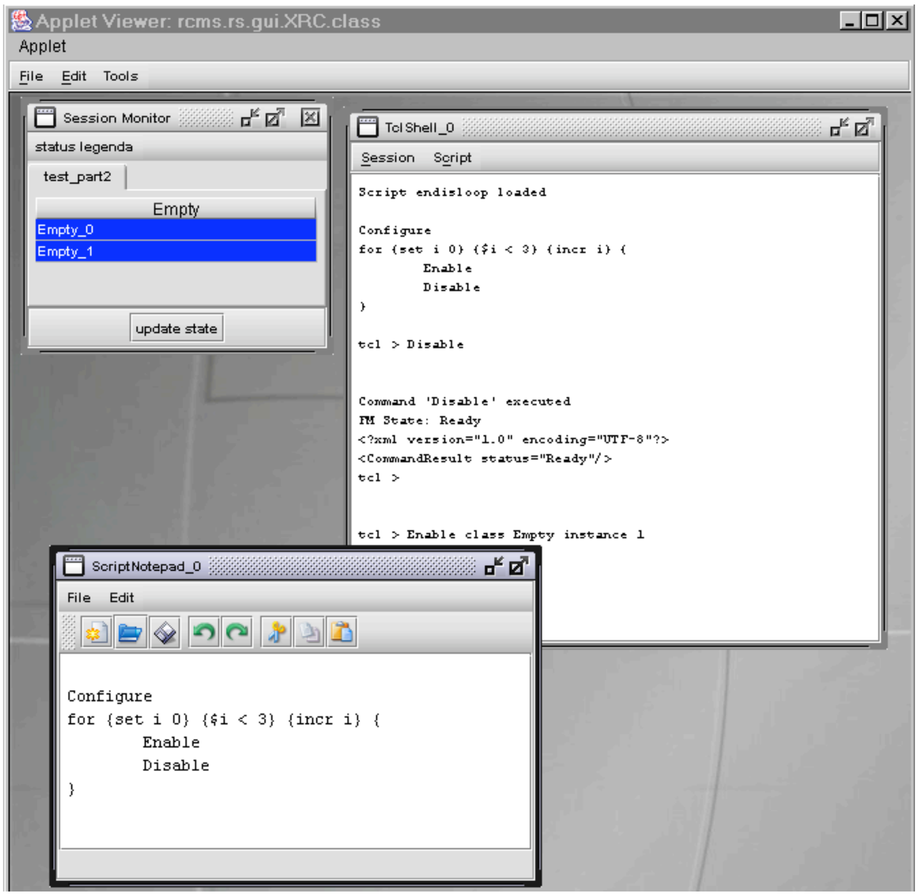
Figure 11: The RunControl applet

4.3.2 The TclShell applet

The **TclShell** applet provides a user interface where tcl scripts or, simply, command line tcl instructions can be used to control and command the system. The tcl scripts can be stored in the RS database by using the **ScriptNotepad** applet. A screen shot of the two applets is shown in figure 12.

The tcl interpreter has been extended with the following commands:

Halt	halt
Reset	reset
Enable	enable
Disable	disable
Configure	configure
Resume	resume
Suspend	suspend
ParameterGet	parameterGet
ParameterSet	parameterSet
UserCommand	userCommand
GetStatus	getStatus
CheckStatus	checkStatus
InterfaceQuery	interfaceQuery
ParameterQuery	parameterQuery
RetrieveApplicationsStatus	retrieveApplicationsStatus



```
Script endisloop loaded

Configure
for {set i 0} {$i < 3} {incr i} {
        Enable
        Disable
}

tcl > Disable


Command 'Disable' executed
FM State: Ready
<?xml version="1.0" encoding="UTF-8"?>
<CommandResult status="Ready"/>
tcl >


tcl > Enable class Empty instance 1
```

```
Configure
for {set i 0} {$i < 3} {incr i} {
        Enable
        Disable
}
```

Figure 12: The TclShell and the ScriptNotepad applets

In order to use the two applets some libraries need to be copied locally to the JDK1.4 installation of the host where you are running the GUI.
Go to the directory **RCMSDir**/lib/auxiliary

```
> cd RCMSDir/lib/auxiliary
```

The libraries **tcljava.jar**, **jacl.jar** and **dom.jar** must be copied in the directory

```
JAVA_HOME/jre/lib/ext
```

of the client JDK1.4 installation.

Unfortunately while a tcl script is running the GUI freezes!
A possible workaround is to open two GUIs, one used for scripts and another one for the other functionalities.
Another possibility is to use the tcl scripting facility from a non-graphical Java application, as explained in the following paragraph.

4.3.3 The Tcl application

The Tcl application provides the same functionalities as the tclShell applet described in the previous paragraph. It allows also the execution of scripts stored in a file.

Go to the directory

```
    > RCMSDir/gui/src/java/rcms/rclets/tclShell
```

Execute the Java application

```
    > ./clientTclShell.sh
```

5 How to Extend the GUI

The GUI framework can be extended with new java applets accessible by the **Tools** menu.
The distribution provides three applets, described in the previous paragraphs, developed following the rules below.
These applets are located in the directory

```
> RCMSDir/gui/src/java/rcms/rclets
```

The subdirectory **RunControl** contains the code for the RunControl applet.
The subdirectory **tclShell** contains the code for the TclShell applet.
The subdirectory **notepad** contains the code for the ScriptNotepad applet.

Having a look to these applets is the best way to understand how to implement and integrate your applets in the GUI framework.

Create a subdirectory, named for instance **userApplet**, where to put your java applet code. Your applet must inherit from the RClet class.

```
public class UserApplet extends RClet {
  // User applet code
}
```

Then write a property file, **UserApplet.properties**, containing the following entries:

```
# Resource Strings for UserApplet example
title=User Applet
type=control
menuItem=Tools
loadOnStartup=false
allowMultipleInstances=false
```

The **title** flag gives the name of your applet.
The **allowMultipleInstances** flag tells if multiple instances of the applet are allowed.
The other flags have no effects in the present release.

When your applet is ready you need to compile it.
Go to the directory

```
> cd RCMSDir/gui
```

Edit the file **build.xml**.
Add your applet in the “target” rclets

```
<target name=”rclets” depends=”…,userApplet”/>
```

Then add the “target” specific to your applet

```
<target name=”userApplet”>
```

```
    ...
  </target>
```

Now when you compile the RCMS source code also your applet will be compiled and deployed into the jakarta tomcat container.

6 How to Customize the Function Managers

The Function Managers can be customized in order to perform actions specific to your partition.
The default behaviour for a Function Manager is to forward a command to all the XDAQ applications defined in a partition. This behaviour is specified in a class called **XDaqSMAdapter** located in the directory **RCMSDir**/rs/resources.

If you need to change the default behaviour you can write your own java class (**ExampleSM.java**) that inherits from the **XDaqSMAdapter** class.

```
// ExampleSM.java
public class ExampleSM extends XDaqSMAdapter {
  public CommandResult enable () throws XDaqSOAPException {
    String[] enableSequence = { “RU”, “BU”, EVM” };
    xdaqPartition.setEnableSequence( enableSequence );
    return xdaqPartition.enableXDaqApplications();
  }
}
```

The ExampleSM class overrides the **enable** method so that the Enable command is delivered to the XDAQ applications in a predefined order.
The xdaqPartition variable is an instance of the **XdaqPartition** class, located in the directory **RCMSDir**/manager/src/java/rcms/manager/xdaqAdapter.
The XDaqPartition class provides all the information about a Partition, as defined in the Resource Service, and a number of methods to access such information. For more information take a look at the methods it provides.
More examples of user classes are provided in the directory **RCMSDir**/rs/resources.

If you put your user class in the directory **RCMSDir**/rs/resources it will be compiled and deployed when you compile the RCMS source code (see paragraph 3.2.4).

The user class can also be compiled and deployed while the RCMS services are already running into the servlet container.
You don’t need to restart the services.
Go to the directory **RCMSDir**

```
  > cd RCMSDir
```

Compile and deploy all the Function Manager user classes

```
  > bin/ant fsm
```

Your Function Manager user class can now be used. Just remember to set the correct class name when you create or modify your partition through the GUI, and reopen your session.